

# 환경 셋팅

## install Cuda 10.2

```
$ wget  
https://developer.download.nvidia.com/compute/cuda/10.2/Prod/local_installers/cuda_10.2.89_440.33.01_linux  
.run  
$ sudo sh cuda_10.2.89_440.33.01_linux.run  
  
# x server로 인한 error 발생 시만 실행  
# ctrl + alt F1 쳐서 텍스트 터미널 진입 / 나올 때는 ctrl + alt + F7  
$ sudo service lightdm stop  
$ sudo init 3  
$ sudo sh cuda_10.2.89_440.33.01_linux.run  
$ sudo service lightdm start  
# 재부팅 후 설치 완료  
  
# Setup CUDA 10.2 path.  
echo "export PATH=/usr/local/cuda-10.2/bin${PATH}:+:${PATH}" >> ~/.bashrc  
source ~/.bashrc  
  
#확인
```

## Install Anaconda

아나콘다 설치 링크 : <https://www.anaconda.com/products/individual>

```
$ cd ~/Downloads  
$ sha256sum Anaconda3-2019.03-Linux-x86_64.sh #파일 이름 다를 수 있으니 앞 부분만 치고 tab  
$ bash Anaconda3-2019.03-Linux-x86_64.sh  
$ source ~/.bashrc  
  
#가상환경 생성  
$ conda create -n yolov5 python=3.8  
$ conda activate yolov5
```

## Install Yolov5

```
$ git clone https://github.com/ultralytics/yolov5.git  
# if your yolov5 folder is lock (UBUNTU)  
sudo chmod 777 -R ~/yolov5  
  
# update  
conda update -yn base -c defaults conda  
  
# install Lib for YOLOv5  
conda install -c anaconda cython numpy pillow scipy seaborn pandas requests  
conda install -c conda-forge matplotlib pyyaml tensorboard tqdm opencv  
pip install pyqt5
```

```
# install pytorch  
conda install pytorch=1.7.0 torchvision=0.8.1 torchaudio cudatoolkit=10.2 -c pytorch  
  
# Extra  
conda install -c conda-forge onnx
```

## PYQT

### UI 설정

```
# QT designer 사용  
# .ui 파일의 위치 -> ~/yolov5/Mainwindow.ui && ~/yolov5/SettingWindow.ui  
# QMainWindow 상속해서 클래스 생성 시 Argument로 넣어주면 ui 사용 가능  
$ conda activate yolov5  
$ designer
```

### Object Detection on PYQT Window

```
$ python3 detect_on_pyqt.py --source 0 --nosave --weights ~/yolov5/Person_Fire.pt
```

#### class LoadStreams

```
# 영상을 지속적으로 띄우기 위한 While문과 같이 사용  
# Detect 클래스 내부에서 for문에 사용되는 객체 생성
```

#### class Detect

```
# 원하는 사물을 Detection하는 클래스  
# QT의 Thread를 사용해서 화면을 띄우는 것과 Object Detection을 동시에 진행  
# 내부의 run 메서드에서 모델을 불러오고 LoadStream에서 불러온 영상을 신경망으로 넣어 객체를 검출  
# yolov5 코드에서 가져옴
```

#### class MainWindow

```
# PYQT창을 띄우는 클래스  
# initializeUI에서 designer에서 생성한 각 파트의 세부 설정을 진행  
# threadEventHanlder에서 반복문의 역할을 수행
```

#### class SettingPage

```
# MainWindow의 setting 버튼을 눌렀을 때 새로운 창 생성
```

## Main

```
# argparse를 사용해 터미널의 명령에서 인자를 받음  
# PYQT 프로그램을 실행시켜주는 QApplication을 생성  
# 내가 생성한 윈도우 객체를 생성 후 화면에 띄움  
# exec_()를 사용해 프로그램을 실행
```

# YOLO v5 Custom Model

## Dataset 생성

### Person

```
# YOLO에서 제공하는 weight파일에 사용된 COCO데이터셋의 사람 데이터 사용  
# ~/yolov5/download_coco_dataset.py 사용해서 person dataset 다운로드
```

### Fire

```
# https://www.floydhub.com/gaiasd/datasets/d-fire
```

## Training

참고 : <https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data>

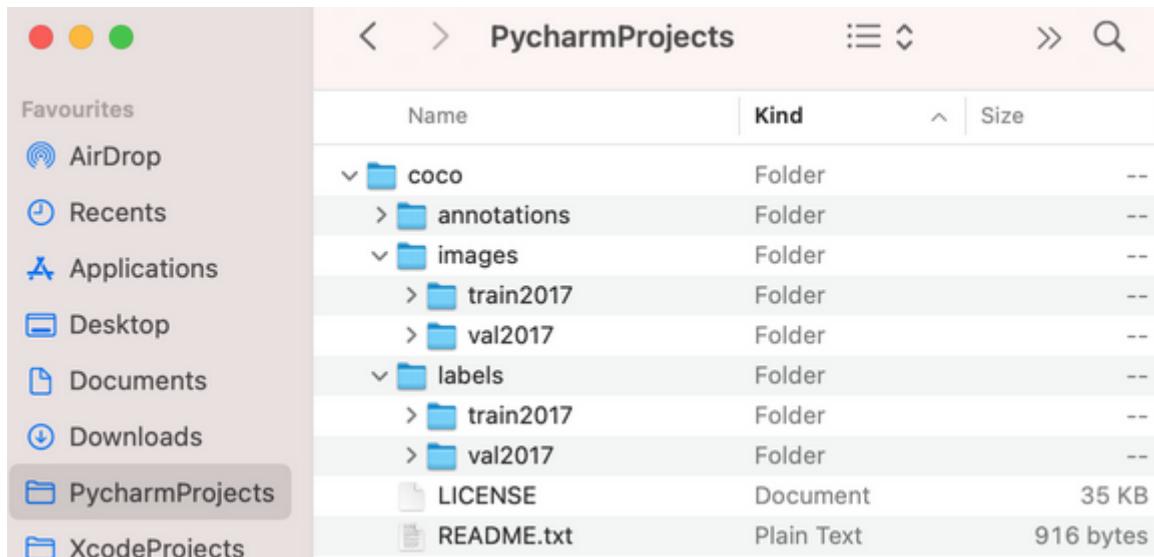
### 1. Yaml

- 학습 데이터 경로, 클래스 개수 및 종류 지정
- train : 학습 데이터 폴더 경로
- val : 학습할 클래스 개수
- nc : 학습할 클래스 개수
- names : 학습할 클래스 이름

### 2. Label

- 라벨된 데이터셋을 가지고 있다면 좋지만 불가피하다면 labellImg 사용
- <https://github.com/tzutalin/labellImg>

### 3. Dataset Location



The screenshot shows a macOS Finder window titled "PycharmProjects". The left sidebar lists "Favourites" including AirDrop, Recents, Applications, Desktop, Documents, Downloads, PycharmProjects (which is selected), and XcodeProjects. The main pane displays a file tree under "PycharmProjects":

Name	Kind	Size
coco	Folder	--
annotations	Folder	--
images	Folder	--
train2017	Folder	--
val2017	Folder	--
labels	Folder	--
train2017	Folder	--
val2017	Folder	--
LICENSE	Document	35 KB
README.txt	Plain Text	916 bytes

- 전체 데이터 폴더 내부에 이미지와 라벨 폴더 존재
- 이미지 폴더 내부에 train과 label 폴더 구분
- 라벨 폴더 내부에 train과 label 폴더 구분

### 4. Train Configuration

--data : data yaml 파일 경로

--weights : 전이 학습 시 Pre-Trained 경로

--batch-size : 배치 사이즈 값(GPU에 맞춰서 하지만 배치는 클수록 과대적합 방지에 좋음)

```
$ python train.py --img 640 --batch 8 --epochs 500 --data disaster.yaml --weights yolov5m.pt
```